Constructional Analysis

John Bryant
Department of Computer Science
University of California at Berkeley
Berkeley, CA 94720
jbryant@icsi.berkeley.edu

Abstract

We describe a unique rule-based language analysis system that performs both syntactic and semantic analysis using construction grammar. The analyzer supports a grammar formalism called the Embodied Construction Grammar (ECG), which is described in the paper along with the computational machinery needed to support such an expressive grammar. We also show how to leverage the rich semantics associated with an ECG grammar. With the rich semantics, the analyzer can place an ordering on competing analyses and can also better respond to unanticipated syntactic patterns. The paper then closes with a description of the analyzer's current applications.

1 Introduction

The *constructional analyzer* is a language analysis system that utilizes a grammar formalism called *construction grammar* to combine traditional syntactic and semantic analysis into one process. A *construction* is a pairing of form and meaning, acting as the cognitively-motivated bridge between the syntax and the semantics. Since form and meaning are inextricably entwined, the constructional analyzer performs syntactic and semantic analysis concurrently, producing a web of syntactic and semantic relations called a *constructional analysis*.

Our approach to language analysis is unique because it is the first to seriously utilize this notion of constructions. Historically, there was a gap between grammar formalisms precise enough to be used in language analysis and those used in the cognitive approaches to language from which construction grammar springs. The Embodied Construction Grammar formalism (ECG) (Bergen and Chang, 2002) was designed to make construction grammar precise enough to be implemented by providing a formalism that marries the precision of unification grammar with the expressiveness of cognitive linguistics.

Since ECG is more expressive than traditional grammar formalisms on both the syntactic and semantic sides, our approach to constructional analysis needs to be more flexible than traditional analysis methods to support all the mechanisms in the formalism. The basic system is an implementation of these mechanisms.

A system that supports so much flexibility also needs to balance that flexibility against the increased computational burden that comes along with it. In a manner analogous to the two poles–the form and meaning–of a construction, our approach to minimizing the computational impact also has two sides, one for the form and one for the meaning. On the form side, the system uses ideas from the partial parsing community for more scalable recognition of syntax. On the meaning side, the system leverages the cognitively-motivated semantics to heuristically rank analyses with a metric we refer to as *semantic density*.

In the next section, we provide a description of ECG. Section 3 further describes how the analyzer works and Section 4 describes two demanding language analysis tasks requiring the semantic capabilities provided by the constructional analyzer.

2 Embodied Construction Grammar

The grammar formalism that makes constructional analysis possible is the Embodied Construction Grammar formalism. ECG combines a grammar formalism and knowledge representation language in a unification-based framework. This allows both constructions and frame-based, schematic knowledge to be expressed succinctly in the same formalism.

As usual for construction grammar, the grammar rules in ECG are pairs, mapping a particular lexical/morphological/syntactic pattern to a (partial) specification of a situation. In ECG, this description of a situa-

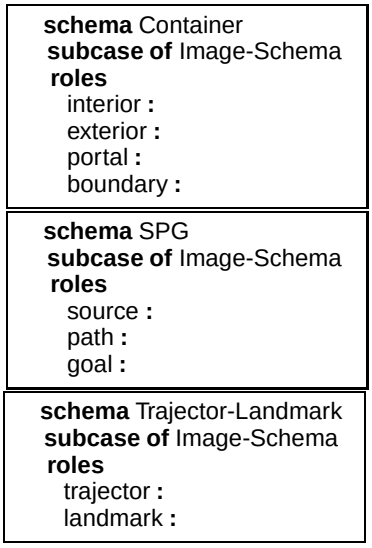


Figure 1: Some image schemas in ECG notation.

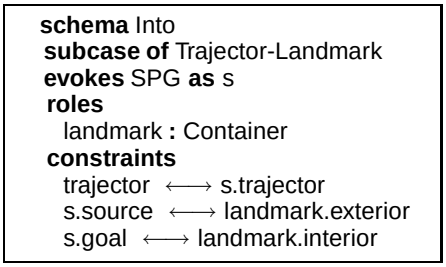


Figure 2: The Into schema.

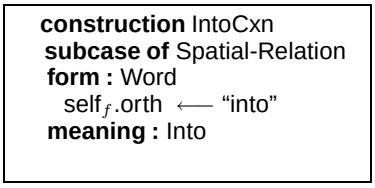


Figure 3: The Into lexical construction.

tion is known as a semantic specification (semspec). The semspec combines embodied semantic primitives like image schemas (Lakoff, 1987) and executing schemas (Narayanan, 1997) with frame-based knowledge (Fillmore, 1982) to completely specify the meaning of an utterance.

Meaning in ECG is represented by *schemas*. Schemas, much like frames, are schematic, role-based conceptual and semantic structures. The schema formalism, however, is augmented augmented by special semantic operators that provide enhanced semantic expressiveness, making it easier to precisely describe insights from the cognitive semantics community.

As an initial starting point into the formalism, figure 1 shows three canonical image schemas in ECG notation. Each schema is initially defined by the keyword **schema**, and after the name, an optional **subcase of** line denotes the structures from which the current schema inherits. Much like frames, ECG schemas (and constructions) are arranged into a type hierarchy In this case, each image schema inherits from the Image-Schema type. Following the **subcase of**, comes the optional roles block, denoted by the **roles** keyword. Just like the roles of a frame, the roles of a schema are the parameters of the concept being described.

These simple image schemas do not show off all of ECG's schema mechanism, however. For a more complete picture, we now focus on the Into schema shown in figure 2. The Into schema subcases the Trajector-Landmark schema, and thus inherits the trajector and landmark roles from its parent. The Into schema further constrains the landmark role by constraining it to be of type Container.

The Into schema also introduces the new **evokes** operator which makes the specified type locally accessible via the given alias. In this case, the evoked SPG schema acts as the background frame, capturing *into*'s notion of motion. This is the primary virtue of the **evokes** operator. It provides a way to place concepts such as *bachelor* into the context of their larger background frames, as described by (Fillmore, 1982).

After the roles block in the Into schema, comes the optional **constraints** block. Constraints act as the semantic glue with the $\leftrightarrow$ identifying its two argument slots. When a set of slots have been coindexed, a proposed filler to any one of those slots must satisfy the restrictions of all of them. In the Into schema, the locally defined landmark.interior[1] role[2] is identified with the evoked SPG's goal role, while the landmark.exterior role is coindexed with the SPG's source role. These constraints schematically describe how the use of *into* suggests motion from outside some container to the inside of that container.

Figure 3 shows the Into lexical construction. Every construction starts with the keyword **construction**, followed by the name of the construction. Then comes the optional **subcase of** keyword that relates constructions to the constructional type system. The ECG version of the Into construction has a form and meaning pole, notated by the keywords **form** and **meaning**.

Constructions type their form and meaning poles. In the case of our Into construction, the form pole is of schema type Word[3] and the meaning pole is of type Into

[1] ECG uses slot chain notation.

[2] The landmark role has an interior role because it was constrained to be of type Container.

[3] Form in ECG is also represented with schemas.

schema. A typed meaning pole indicates that a particular construction denotes an instance of that type. Thus our Into construction claims that the word *into* means an instance of the complex relation described by the Into schema.

The Into construction also exhibits the assignment operator (←). This operator fills a slot with an atomic value. In our Into construction's form pole, the orth[4] feature brought in from the Word schema is assigned the atomic string *into*.

Figure 4 shows the clausal Caused-Motion construction, an example of which is *The basketball player threw the ball into the basket.* This construction has an agent (the player) which acts upon a patient (throwing the ball) thereby moving it along a path (into the basket). Since the Caused-Motion construction links a particular syntactic form, that of a referring expression, a force-application verb, a referring expression and a path to a Caused-Motion-Scene[5], the construction is different from the ones we have covered so far in that it has constituents that are themselves constructions. Thus instead of typing the form block, the form block has constraints relating the constituents.

Each of the construction's four constituents are defined in the constructional block. Each constituent is assigned a local name, and then after the colon, the constructional type is defined. If necessary, like in the case of the Verb constituent, a semantic type is added in brackets.

The ordering of these constituents is specified by adding form constraints to the form block. When the constructional analyzer searches for instances of a construction, these form constraints must be satisfied. The two supported constraints are **before** which requires that the left argument be somewhere the left of the right argument in the input, and **meets** which requires the left argument to be directly before the right argument in the input.

In the Caused-Motion construction, the form constraints require that the agent be directly before the verb and the verb be before the path and patient. Notice that the relative order of the path and patient is left unspecified[6]. Because ECG allows constituent order to be unspecified like this, ECG can express with one rule what a CFG might have to express with an exponential number of rules.

The meaning pole of the construction uses the semantic machinery that has already been described. It links the agent_m's category to the agent of the scene as well as setting patient_m.category to the trajector of the specified path. Notice that the constraints use the *m* and *f* sub-

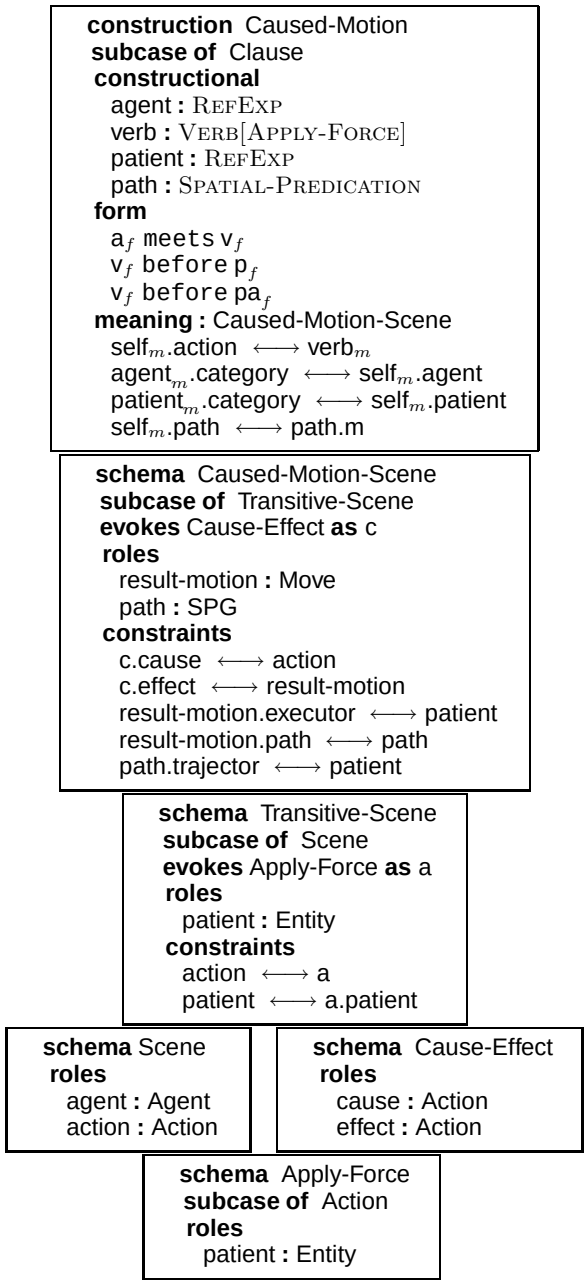


Figure 4: The Caused-Motion Construction and related schemas.

[4]Orth is short for *orthography*.

[5]A caused motion scene is one where the agent applies force to the patient resulting in a shift in the position of the patient.

[6]This might be a partial solution for dealing with what are called *heavy NPs* briefly described in (Bryant, 2003).

scripts when referring to the constructional constituents' form and meaning poles, respectively, and can be applied to any construction as if they were just dotting into the structure.

With a formal language for describing constructions, many avenues are opened. The most important for the sake of this work, is that it is precise enough to be implemented. More specifically, ECG's unification-based lineage makes it possible to translate ECG into the feature structures that are manipulated by the constructional analyzer.

3 The Constructional Analyzer

The approach to constructional analysis we describe in this report uses a level-based syntactic processing model (Abney, 1996). Level-based processing models are more efficient at recognizing syntax because they sacrifice full recursion in the grammar. Each grammar rule is assigned a level and can only take constituents from the levels below it. As a consequence, the analysis task is decoupled a into a chain of smaller analysis tasks.

Such a model was exploited by systems like FASTUS (Hobbs et al., 1996) for efficient extraction of shallow semantic information. But instead of doing shallow semantic analysis for the purposes of information extraction, the constructional analyzer utilizes both the semantic richness of construction grammar and extended computational mechanisms to do full constructional analysis, resulting in both a complete syntactic analysis and a deep semantic analysis. The constructional analyzer integrates the following computational capabilities to make deep analysis possible.

- Support for unification
- Support for multiple concurrent analyses with a chart
- Support for the more flexible form relations found in ECG
- Support for ECG's semantic expressiveness
- An analysis evaluation heuristic
- A method for extracting meaningful semantics from partial analyses

3.1 The Basics of the Constructional Analyzer

Since ECG is a unification-based formalism, supporting unification is a necessary first step. Along with unification, a chart is employed by the system to keep track of the many possible analyses generated during rule-based language analysis.

On the form side, the analyzer cannot use finite state machines or even context free grammars to do matching because of ECG's more relaxed notion of form. ECG does not require the grammar writer to specify a total ordering on a construction's constituents, and thus recognizers requiring a total ordering (like CFG parsers) in each rule are unusable. The constructional analyzer instead uses a computational unit called a *construction recognizer*

A construction recognizer is a chunk of active knowledge into which a construction gets transformed. Each construction recognizer is designed to check both the form and meaning constraints of just the construction it models. In other words, instead of a monolithic parser that takes the grammar rules as input, the constructional analyzer itself is a collection of active construction recognizers working together to generate the constructional analysis.

3.2 Making The System More Robust

Given the mechanisms just described, we already have an approach that will take ECG grammars and generate constructional analyses for every utterance supported by the grammar. Although, when it comes to building a real system, this is not enough. This section describes extensions built into the constructional analyzer to better deal with the challenges associated with trying to analyze real language.

Once such challenge language analyzers are faced with is robust behavior when encountering a well-formed and meaningful utterance not licensed by the grammar. Since it is very hard for linguists to try and anticipate *every* possible meaningful utterance, frequently well-formed language falls through the grammatical cracks.

The ambiguity of language poses another challenge because a particular utterance can be analyzed in so many different ways, that later stages of analysis cannot process them all to find the most appropriate. Thus heuristic methods for indicating which analyses are more likely to be correct can help focus later stages of processing.

Stepping up to these challenges requires a blend of computational mechanism and linguistic insight. The use of the level-based approach is one such example of this blend, but the deep semantics is the real key to making headway. Given the early pairing of form and meaning afforded by constructions, an analyzer with access to such semantics can leverage them to better respond to unanticipated utterances and highlight analyses more likely to be correct.

3.2.1 Leveraging The Semantics

The key insight behind our approach to making the system more robust is the realization that every utterance is trying to communicate a scene. This common scene that the referents in the utterance participate in is parameterized into frames and image schemas in ECG. Now as-

Commercial-Event 1	
buyer :	Harry
seller :	Bill
goods :	a car
price :	

Commercial-Event 2	
buyer :	Harry
seller :	Bill
goods :	a car
price :	1500

Figure 5: The semantic density metric used to compare two semspecs each containing a Commercial-Event frame. The Commercial-Event frame on the left has a semantic density score of .75 and the Commercial-Event on the right has a score of 1. Thus the second frame would be considered better because more of the frame elements are filled in.

suming that a better analysis is one that better describes the scene, one way to compare analyses is by how complete each parameterization is. Those analyses that fill out more of the parameters would be preferred to those that filled out fewer parameters. This is the motivation for the ranking heuristic that we call *semantic density*.

Semantic density compares constructional analyses based upon their semantic content. Analyses that have a higher ratio of filled slots to total slots in their semspec are considered better analyses according to semantic density[7]. A metric like semantic density not only provides the analyzer with a way to focus on the more complete analyses at the end of the analysis process, but it also makes early pruning a possibility. Figure 5 shows a simple example of the semantic density metric in use.

3.2.2 Partial Analyses

In the case of a missing construction, the analyzer is faced with many partial analyses that span some of the input, but no constructional root that spans all of the utterance. In this scenario, the constructional analyzer must salvage meaning from the constituent pieces of some non-existent construction. All is not lost though, since the analyzer has stored all recognized constituents in the chart. Given these constituents, the analyzer can try collections of spanning, non-overlapping constructs, and see how well they fit together[8].

We can precisely define how well the collection of constructs fits together by looking at the collection's semspecs. After grouping their semantic structures, the analyzer performs *structure merging*[9] For every evoked

0. Lexical constructions
1. Noun noun compounds
2. Adjectives
3. Referring expressions
4. Spatial Predications
5. Clausal constructions

Figure 6: The levels used in the example analysis.

structure, if there is another schema that has the same type and is unifiable, then those structures are merged into one structure holding their combined semantic content. This phase in the constructional analyzer simulates the bindings that might *hypothetically* have occurred had there been a construction to license this collection of constituents.

To minimize the number of partial analyses that structure merging is performed on, preference is given to partial analyses that span with the fewest number of constructs. Out of the remaining candidates, how well a particular set of constructs fits together can be measured with the semantic density metric. Because the system compares analyses with a semantic metric, comparing partial analyses is no different than comparing complete analyses.

By this point, we have laid out the design of an analyzer that not only supports the richness provided by constructions, but also uses mechanisms to turn the "burden" of semantics into a virtue by leveraging the semantics in unique ways. At least at the conceptual level, such a system would not only be robust, but semantically rich enough for deep, simulation-based understanding as described in (Bergen and Chang, 2002). Such a system would be appropriate for any environment where deep understanding is a necessity.

3.3 An Example

In order to make the previous discussion more concrete, let's analyze an example sentence using the Caused-Motion construction described earlier. The sentence we will consider is *The basketball player threw the ball into the basket.* Given a grammar that can handle simple referring expressions of the form (Det) Adj* Noun+ (making sure to add the appropriate semantics) and spatial phrases, we can arrange the rules into levels (see figure 6) and generate analyses that use the Caused-Motion construction.

Figure 7 shows an example noun-noun compound construction used in the analysis that puts constituents of

[7]This, of course, is not the only possible metric. For example, one might also want to consider the number of bindings, the number of schemas in each semspec, or the total complexity of construal. If a large corpus of semspecs were available, one could use machine learning techniques to decide how to combine these metrics.

[8]Obviously, the search strategy used for choosing which constructs to try affects the complexity of this process.

[9]FASTUS used structure merging as well, but it was used for resolving structures across utterances, not explicitly within a single utterance.

construction Noun-Noun-Compound
subcase of Category
constructional
a : CATEGORY
b : CATEGORY
form
a_f meets b_f
meaning
$self_m \longleftrightarrow b_m$

Figure 7: A Generic Noun-Noun-Compound construction that just sets the meaning of the construction as a whole to be that of the second constituent. It relies on structure merging to infer the correct relation between the two Category constituents.

type category[10] together to generate an instance of type Noun-Noun-Compound which is itself subtype of category. Thus the rule is recursive with itself and any other category-requiring rule. Notice that it is on the same level that all other category constructions are assigned. The constructional analyzer allows the constructions assigned to the same level to be mutually recursive.

After the Category constructions are processed, simple referring expressions are generated. After the referring expressions, Spatial-Predication constructions are recognized on the next level, and the constructional analyzer is finally ready to match the Caused-Motion construction. Figures 8 and 9 (and the rest of this section) describe the matching process schematically.

In frame A of figure 9, the construction recognizer is in its initial state and it has not found any of the constituents for the Caused-Motion construction. In B, the recognizer has found a referring expression corresponding to *the basketball player*, and since it unifies with the agent role of the Caused-Motion construction, it is accepted as the first constituent. Notice how the node in the graph corresponding to the agent is removed indicating that it has been found.

In frames C and D, the same scenario takes place except it is the verb *threw* and the referring expression *the ball* that satisfy the form and meaning requirements of their corresponding constituents. Notice in C that the construction recognizer is now allowed to find either the patient or the path since both nodes have no incoming edges. In E, we see a completely matched Caused-Motion construction with a complete Caused-Motion scene and an empty constituent graph indicating that a complete instance of this construction has been found.

In short, the construction recognizer builds up a graph data structure to keep track of the constituents and an in-progress semspec to keep track of the semantics. Each constituent that satisfies the form and semantic constraints updates the constituent graph and the in-progress partial semspec. The final result for a successful match has the agent of the caused motion scene to be the player, the patient being the ball, and the goal of the path being the interior of the basket.

[10] An instance of the category construction can either be what is usually considered a noun like *dog* or a noun-noun compound like *vinyl siding salesman* or *gas meter turn-off valve*.

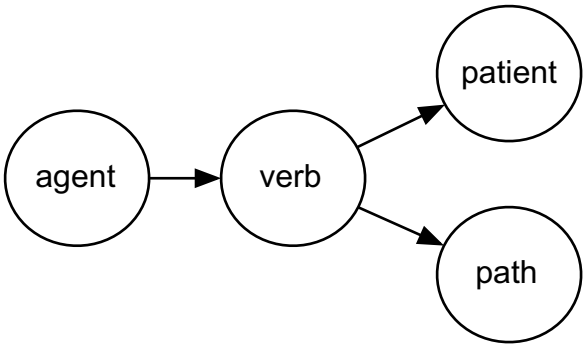


Figure 8: The constituent graph structure for the Caused-Motion construction. Each constituent corresponds to a node in the constituent graph. At any particular point, the construction recognizer is only allowed to search for constituents with no incoming edges. When a constituent is found, its node is removed from the graph along with any outgoing edges from that node. After removing a node, the construction recognizer is now allowed to search for different, newly-released constituents or if there are no nodes left, then a valid instance of the construction (at least with respect to the form constraints) has been found.

3.4 Partial Analyses

While we have seen how the system when works when every construction necessary for an analysis has been defined, but how can the analyzer salvage analyses when all the necessary constructions have not been defined. Consider the case in which the Caused-Motion construction itself had not been present. In such a scenario, if the semantics were appropriately encoded in the grammar, it might still be possible to generate the correct inferences.

Figure 10 shows the lexical construction and schemas associated with *threw*. The construction evokes the default scene that a throw action is associated with. The Caused-Motion scene is a parameterization of a scene in which an agent applies some force to a patient, sending that patient along a path. Attaching the semantics to the verb using the **evokes** operator allows the verb to be connected to a particular argument structure, but without requiring that the verb appear only in that argument structure[11].

[11] This approach to lexical semantics is consistent with a con-

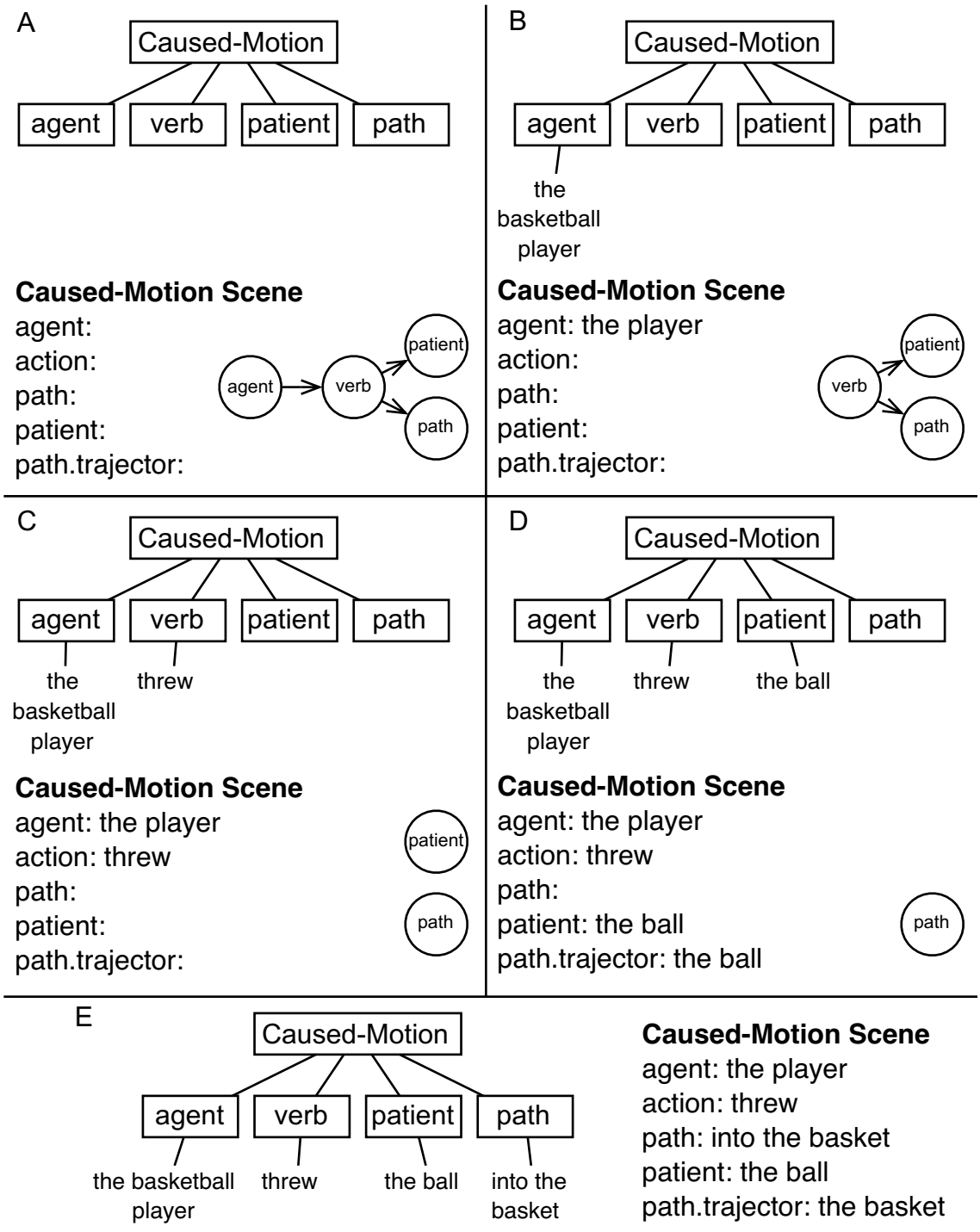


Figure 9: Snapshots of the internal state of the Caused-Motion construction recognizer on the sentence *The basketball player threw the ball into the basket.*

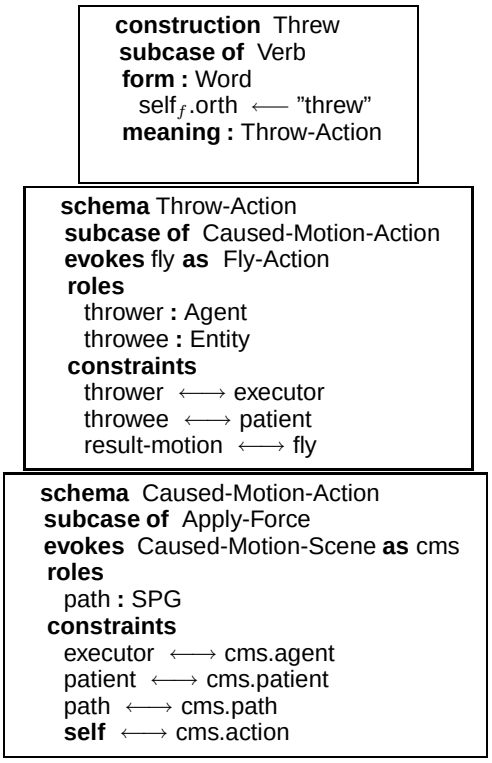


Figure 10: The *threw* lexical construction and associated schemas. This construction defines the meaning of a verb to be the Throw-Action x-schema.

Figures 11 and 12 show the combined semspecs after the Spatial-Predication has been matched, but before the Caused-Motion construction is activated. Even before encountering the Caused-Motion construction, however, the analyzer could have made sense of this semspec. This is because the analyzer already has an idea about which semantic complements that *threw* takes because its lexical construction indirectly evokes the Caused-Motion-Scene schema.

When the evoked Caused-Motion-Scene was added to the semspec (by the Throw-Action schema that is the meaning pole of the Threw construction), it added dummy place-holder schemas for the agent, patient, and path roles. These schemas also have the property of being evoked because the schema they belong to is evoked. Now if the analyzer merged unifiable structures, the agent's dummy schema would be unified with the Player schema, the patient's dummy schema with the Ball schema, and path's dummy SPG with the SPG brought into the semspec by the Spatial-Predication. This is exactly what the analyzer does, as shown in figure 13.

The ball could not have been unified with the agent because it does not have agentive properties (without some kind of construal). Further the Basketball schema could have been unified with the patient, but that would be dispreferred because at least one of the syntactically licensed referents would not have been a part of the scene at all.

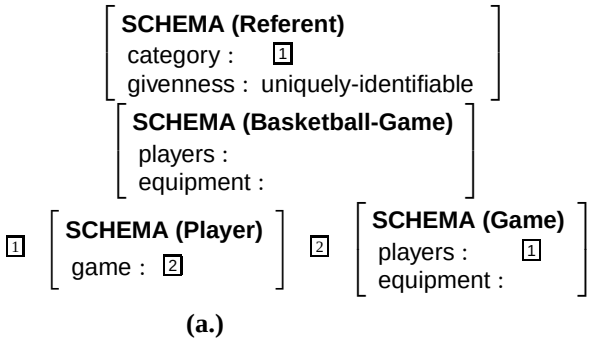
Also notice the unification of the evoked Game and Basketball schemas that were brought in by *The basketball player* phrase. This simple structure merging mechanism successfully merged the background frames, leading to the correct semantics. i.e. that the *player* was a player of *basketball*.

Such a simple mechanism for inferring semantic roles can be very helpful at generating meaningful analyses when the syntactic coverage is lacking. The primary requirement on the grammar writer is for the lexical items to be imbued with enough semantic information to make inferences such as these possible.

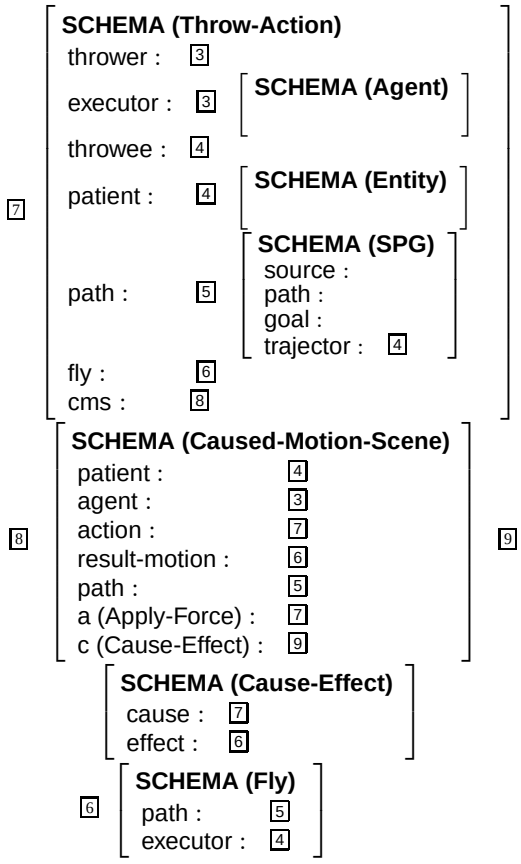
The inferential mechanism is not perfect, though, as it could lead to incorrect inferences. Consider the sentence *The linebacker pushed the offensive tackle into the quarterback.* Without the Caused-Motion construction to link the grammatical subject of the sentence to the agent, one possible inference that the analyzer might make is that the patient was the linebacker and the agent was the offensive tackle, in effect assuming that it was analyzing a language where patient comes before agent[12]. Of course

struction grammar approach to argument structure, as it allows the meaning of a verb to be the combination of its semantics and the semantics of the construction it appears in. For more information see (Goldberg, 1995).

[12]Remember that this inference did not happen in the last example because a ball does not generally have agentive proper-

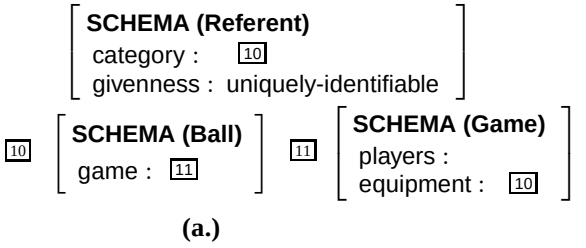


(a.)

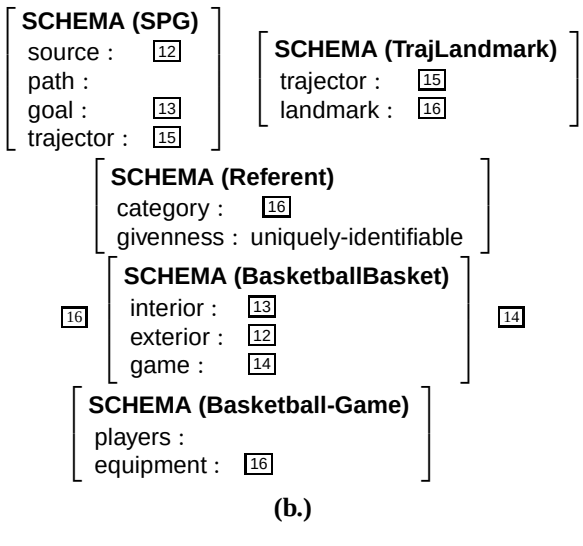


(b.)

Figure 11: The semantic structures associated with *The basketball player* in (a.) and *threw* in (b.). The Referent schema in (a.) is the meaning of the whole referring expression. The single Throw schema in (b.) is the meaning of the verb.



(a.)



(b.)

Figure 12: The semantic structures associated with *the ball* in (a.) and *into the basket* in (b.). The Referent schema in (a.) is the meaning of the whole referring expression, and the SPG in (b.) is the meaning of the spatial phrase.

it would also generate an alternative analysis corresponding to the correct scenario, but it would have not way of preferring one over the other[13].

4 Applications

The constructional analyzer is currently being put to use in two tasks that require the deep semantics that it provides.The first task is that of *Simulation-Based Language Understanding* (Narayanan, 1997), and the second is the task of inductive learning of constructions (Chang and Maia, 2001).

4.1 Simulation-Based Language Understanding

Simulation-based language understanding draws inferences about actions and events by executing an active model of actions and events. The active models are called *x-schemas* (short for executing schemas) which are extensions to stochastic petri nets. In order to draw inferences about an action like walking, the system performs a *simulation* by executing its internal model of walking.

To tailor the inference to a particular scenario, the simulator needs to set the *parameters* of the x-schema representation appropriately. These parameters are the free variables that control how the simulation executes. For example, the walking x-schema would have parameters for who the walker is and for the path (or direction) of motion. So in the sentence *Harry walked into the cafe*, the walker would be *Harry* and the path would be *into the cafe*.

But before a language understanding system can utilize the virtues of such a model, the parameters must be extracted from the utterance. This is where the constructional analyzer comes in. If the constructions have features designed to interact with the simulator, each constructional analysis will provide the parameters to the simulation, and the analyzer and simulator will interact to understand the utterance.

At the time of this writing, researchers are working to integrate the constructional analyzer with a simulation engine. Once coupled, the constructional analyzer and simulation engine will be a unique and powerful method for language understanding. Such a system would even be able to do "exotic" metaphorical inference with ease, since metaphorical inference is just a special kind of parameterization.

ties.

[13] A similar situation happens for an entity trying to learn language. First it would consider both possibilities, but it would have the ability to rule one out given the observation of the actual scenario. Unfortunately, the analyzer does not have access to the scenario that the utterance is describing. See (Chang and Maia, 2001) for a discussion of a language learning algorithm that does have access to the scenario.

4.2 Child Language Learning

The language learning model used by Chang (Chang and Maia, 2001) is a comprehension-based model of language learning built on the following assumptions:

- There is significant prior knowledge going into the learning process.
- The learning is incremental and based on experience.
- The learning is tied to language use. i.e. Frequency of language data affects the learning.

The model analyzes the incoming utterance using the current set of constructions that have been learned. If the current analysis generated by the constructional analyzer cannot explain all of the semantic content found in the current scenario associated with the utterance, the model hypothesizes new constructions. This hypothesis process pairs up the unused form relations with the missing semantic relations to produce constructions that fill in the semantic/pragmatic gap. Since the hypothesis process is under-constrained, the model generates multiple constructions in an attempt to explain the same missing semantic content. The more useful of these constructions in later analyses are the ones that get reinforced while the others wither away.

This model of learning depends on the language analyzer to initially try and explain the semantics of the scene. But for such an analyzer to be useful it needs to be semantically focused. It also needs to be capable of incremental analysis as well as tolerant of noise and missing constructions. These requirements line up perfectly with the constructional analyzer's virtues primarily because the language learning task heavily influenced the design of the constructional analyzer. In effect, the constructional analyzer was built on the assumption that all grammars, not just grammars in the process of being learned, will lack coverage when faced with real language.

5 Conclusion

This report describes the constructional analyzer, an implemented analysis system for producing constructional analyses. Such a system is important not only because it is the first implemented analyzer of a formalized construction grammar, but also because of its broader implications on computational linguistics.

One of the most striking ways this work differs from other approaches is its reliance on knowledge. In other words, the constructional analyzer produces relational structures. This goes against the currently in fashion practice of producing extremely limited, but statistically

well-motivated structures. Ideally, the statistical motivation should go hand in hand with relational structure, providing the best of both worlds. This work takes those initial steps back toward knowledge, making possible the transition to robust probabilistic relational models[14].

Beyond the knowledge based approach, another separate question emerges, that of which knowledge should be represented. This work describes an approach using embodied semantics. Thus this work is important because it provides computational underpinning for a vast body of work in cognitive science, finally making it possible to show the advantages of such an approach using real applications. Two important applications are grammar learning and simulation-based understanding, both of which could change the way people build language understanding systems.

The final implication of this work is that it is arguably more scalable and robust than other constraint-based systems like HPSG (Pollard and Sag, 1994). The reasons for this are the combination of using a constructional approach to language analysis, using the deep embodied semantics as the semantic substrate, and then leveraging that semantics using metrics like semantic density. Constructions make the system more scalable and robust because they free the grammar from having to deal with everything lexically. The deep semantics provides scalability because of the clear interfaces between the analyzer and the simulation engine. Furthermore, having access to the semantics not only rules out semantically malformed analyses, but also provides means for choosing the heuristically best analyses. Both of these make the system scale better and provide robustness in the face of unanticipated language.

References

Steven Abney. 1996. Partial parsing via finite-state cascades. In *Proceedings of the ESSLLI '96 Robust Parsing Workshop*.

Benjamin Bergen and Nancy Chang. 2002. Embodied construction grammar in simulation-based language understanding. Technical Report TR-02-004, ICSI. To appear in Oestman and Reid, eds., Construction Grammar(s): Cognitive and Cross Lingusitic Dimensions. John Benjamins.

John Bryant. 2003. Constructional analysis. Master's thesis, UC Berkeley.

Nancy Chang and Tiago Maia. 2001. Learning grammatical constructions. In *Proceedings of the Conference of the Cognitive Science Society*.

[14]See (Pfeffer, 1999) for a description of such models.

Charles Fillmore. 1982. Frame semantics. In *Linguistics in the Morning Calm*, pages 111–138. Linguistics Society of Korea.

Adele Goldberg. 1995. *Constructions: A Construction Grammar Approach to Argument Structure*. University of Chicago Press.

Jerry Hobbs, Douglas Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson. 1996. Fastus: A cascaded finite-state transducer for extracting information from natural-language text. In Roches and Schabes, editors, *Finite State Devices for Natural Language Processing*. MIT Press.

George Lakoff. 1987. *Women, Fire, and Dangerous Things.* University of Chicago Press.

Srini Narayanan. 1997. *Knowledge-Based Action Representations for Metaphor and Aspect*. Ph.D. thesis, University of California at Berkeley.

Avi Pfeffer. 1999. *Probabilistic Reasoning for Complex Systems*. Ph.D. thesis, Stanford University.

Carl Pollard and Ivan Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

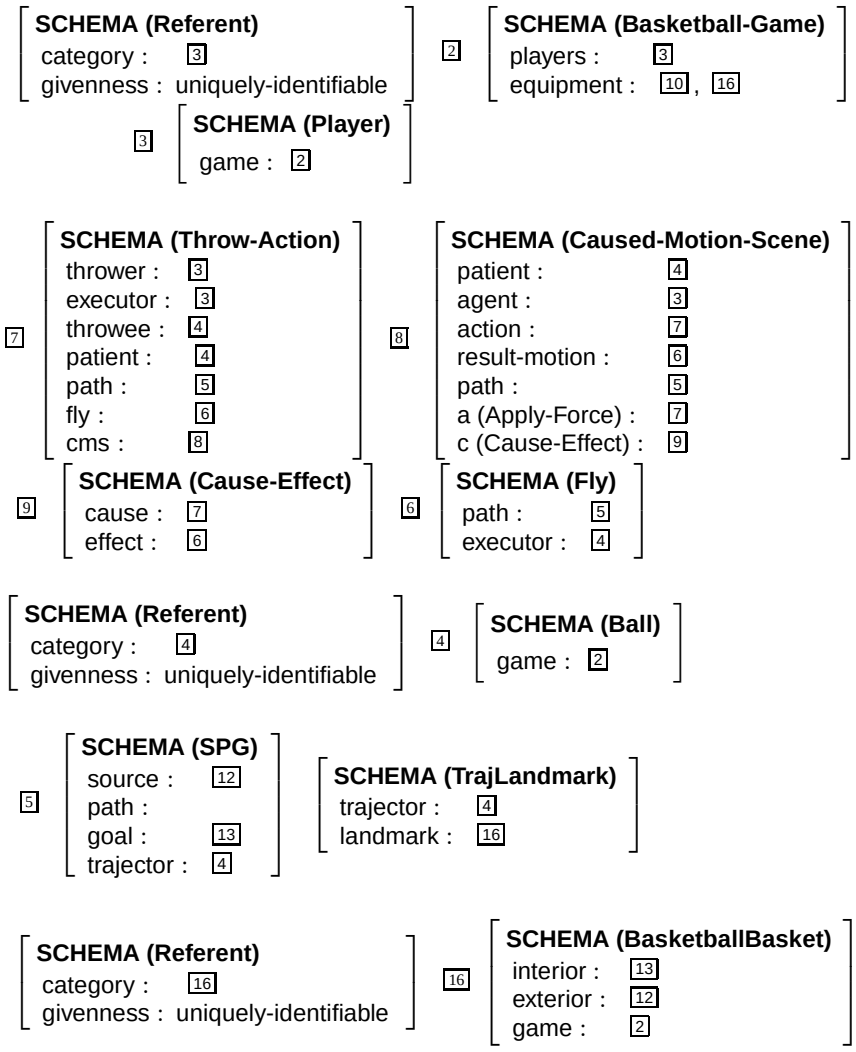


Figure 13: The collection of schemas belonging to constituents spanning *The basketball player threw the ball into the basket* but before the Caused-Motion construction is activated, except now evoked structures have been merged.